Logica Dance Of The Bees

Door: Thijme de Bruijn

Groep: 2

# Inleiding

In dit document staat alle logica dat gebruikt gaat worden voor het project ‘The Dance Of The Bees’ door groep 2 voor het eindproject van ADB. Voor dit project hebben wij predikaatlogica gebruikt. Om het overzichtelijker te maken, hebben we de logica onderverdeelt in vakjes. Er zal eerst uitgelegd worden wat we willen wat er wordt gedaan en daarna wordt de lijn logica gedeeld. De meeste logica lines (en sws alle logica) wordt gebruikt en kan teruggevonden worden in de code.

## Movement

De logica van wanneer een tile naast een andere tile in het grid zit.

∀a ∀b ((Tile(a) ˄ Tile(b) ˄ (Equal(xas(a), xas(b)) ˄ (Equal(MinusOne(yas(a)), yas(b)) ∨ Equal(PlusOne(yas(a)), yas(b)))) ∨ (Equal(yas(a), yas(b)) ˄ (Equal(MinusOne(xas(a)), xas(b)) ∨ Equal(PlusOne(xas(a)), xas(b))))) -> Neighbours(a, b))

Een bij kan alleen bewegen naar een vakje naast hem op het grid.

∀a ∃b ∀c ((Bee(a) ˄ Tile(b) ˄ Tile(c) ˄ Neighbours(a, b) ˄ WantMove(a)) -> Move(a, b))

## Bij

Wanneer een bij, een bloemenveld of een bijenkorf op hetzelfde vakje staan, raken ze elkaar aan.

{\displaystyle \forall }{\displaystyle \forall }∀a ∀b ∃p ((PhysicalConcept(a) ˄ PhysicalConcept(b) ˄ Tile(p) ˄ (IsAt(a, p) ˄ IsAt(b, p)) -> Touch(a, b)))

Als een bij geen nectar heeft en op een bloemenveld staat die wel nectar heeft, pakt hij een nectar van dat bloemenveld af.

∀a ∀b ((Bee(a) ˄ FlowerField(b) ˄ ¬ HasNectar(a) ˄ (Touch(a, b) ˄ Nectar(b) ) -> (GainNectar(a) ˄ LoseNectar(b)))))

Wanneer een bij nectar heeft, wil hij naar huis gaan.

∀a ∃k ((Bee(a) ˄ Beehive(k) ˄ HasNectar(a)) -> GoHome(a, k)))

Wanneer een bij de volledige grid heeft bezocht, wil hij naar huis gaan.

∀a ∃g ∃k ((Bee(a) ˄ Beehive(k) ˄ GridExplored(g)) -> GoHome(a, k)))

## Aankomst bij de korf

Wanneer een bij op zijn korf staat, verliest hij de nectar die hij heeft en geeft hij deze nectar af aan de bijenkorf.

∀a ∃k ∃f ((Bee(a) ˄ Beehive(k) ˄ FlowerField(f) ˄ Touch(a, k)) -> (GainHoney(k) ˄ LoseNectar(a)))

Wanneer een bij op zijn korf staat, verliest hij zijn grid memory.

∀a ∃k ∃f ((Bee(a) ˄ Beehive(k) ˄ FlowerField(f) ˄ Touch(a, k)) -> LoseGridMemory(a))

Wanneer de bij terugkomt in de korf, krijgt hij een clue waar hij een bloemenveld kan vinden.

∀a ∃k ∃f ((Bee(a) ˄ Beehive(k) ˄ FlowerField(f) ˄ Touch(a, k)) ˄ HasNectar(f) -> GainClue(a, f)

Wanneer een bij de korf verlaat, neemt hij een beetje honing van de korf die hij gebruikt om het energielevel van de bij volledig aan te vullen.

∀a ∃k ((Bee(a) ˄ Beehive(k) ˄ (Touch(a, k) ˄ HasHoney(k) ˄ LeavesHive(a)) -> (GainEnergy(a) ˄ LoseHoney(k))))

## Bij energie

Aan het einde van de tijdstap verliest een bij een energie.

∀a ∀b ((Bee(a) ˄ TimeStep(b)) -> MinusOne(Energy(a)))

Wanneer een bij geen energie meer heeft, gaat hij dood.

∀a ((Bee(a) ˄ IsZero(Energy(a))) -> Die(a))

Wanneer een bij net genoeg energie heeft om naar huis te gaan, gaat hij naar huis.

∀a ∃k ((Bee(a) ˄ Beehive(k) ˄ NeedToGoHome(TicksToHome(a, k), TicksOfEnergy(a)) -> GoHome(a, b)))

## Bloemenveld

Een bloemenveld groeit om de zoveel tijd nectar terug.

∀a ((Flowerfield(a) ˄ IsZero(NectarRegrowCountdown(a) -> (GainNectar(a) ˄ StartNectarRegrowCountdown(a)))

Wanneer een bij een bloem ziet, gaat hij daarop af.

∀a ∀b ∃g ((Bee(a) ˄ FlowerField(b) ˄ ¬ Touch(a, b) ˄ Sight(a, b, g)) -> GoToFlowerfield(a))

### Hivemind

Om de zoveel tijdstappen, wordt de grid memory van iedere bij met elkaar gedeeld.

∀a ∀b ∃c ((Bee(a) ˄ Bee(b) ˄ Model(c) ˄ IsZero(HiveMindCounter(c)) -> HiveMind(a, b))

Wanneer er een hivemind signaal wordt gestuurd, delen alle bijen hun grid memory.

∀a ∀b ((Bee(a) ˄ Bee(b) ˄ HiveMind(a, b)) -> (GainGridMemory(a, b) ˄ GainGridMemory(b, a)))

Wanneer het hivemind signaal wordt verstuurd, wordt er onder de bijen overlegd wie naar welk bloemenveld gaan.

∀a ∀b ((Bee(a) ˄ Bee(b) ˄ HiveMind(a, b)) -> Auction(a, b))